Mind Map Generator

[1]Arti Koul, [2]Rachit Patani, [3]Soumya Prakash Dasmohapatra, [4]Prof. Prachi Tawde

Information Technology, Dwarka Das J Sanghvi College of Engineering, Mumbai, India

Abstract: We are currently living in the so-called information age generating huge volumes of data every second. With such huge volumes of data generated every second the process of going through the entire data and extracting relevant information out of it is tedious. This is where mind maps can be useful as an effective knowledge representation tool. Mind-mapping is a process in which we draw a diagram in a way similar to how our brain visualizes the information. A common problem that arises when people draw these mind-maps manually is that there is no way to verify the correctness of the drawn mind-maps. Moreover, this process is very time-consuming and requires a lot of patience and stamina from the person who is drawing these mind-maps. Therefore, the need to generate mind maps automatically from text arose. This is the main aim of our project.is to automatically generate mind-maps directly from the text given as input using natural language processing and text-mining algorithms. We have developed a full-featured web-application where the user can directly put text and generate the corresponding mind-map of the text.

Keywords: Mind maps, text-mining, algorithms, web-application.

1 Introduction:

It was in 1452 that Johannes Gutenberg invented the movable type to make writing easier. After that, the number of documents having printed text has exponentially increased. Moreover, as the world becomes more and more digitized by the day the amount of textual data generated per second is massive. A survey was conducted by [1] that reported at every second around 1.7 megabytes of data is generated out of which a majority is in an unstructured textual format. As the data is unstructured, it is very difficult to extract information that is useful to us on time. The process is very time-consuming and inefficient and this goes against the main objectives of digitization i.e., to deliver such products that can be used to reduce effort, energy, and time. Furthermore, thinking from a company's perspective their knowledge bases also have huge volumes of information that need to be properly used to enhance the productivity of the company. Therefore, keeping this in mind a method called mind-mapping was proposed by Tony Buzan. Our brain is not able to actively retain information when text is given in a linear format thereby preventing the brain from making associations reducing creativity and memory. This is where mind-mapping comes into the picture. Mind-map is a structure that unleashes the brain's true potential as in a mind map there is a central keyword (This keyword is the main topic of the entire data) which is connected to the concepts that are related to it graphically. A mind map consists of series of nodes that form a hierarchy where each node depicts a particular concept These concepts are connected to depict a relationship between them [2].

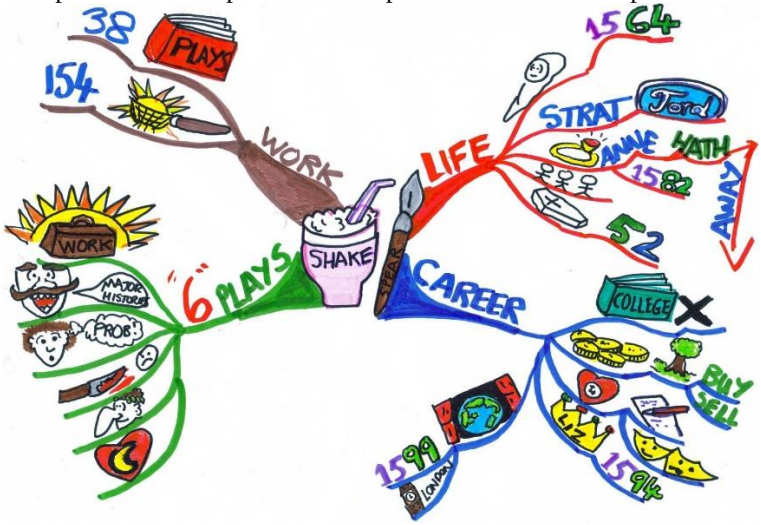

Figure 1 Mind Map

Figure 1 shows the mind map of Shakespeare. The central keyword here is Shakespeare. After that four sub-concepts are linked to the central keyword. The four subconcepts are work, life, career, and his six plays. These sub-concepts are further branched out to depict relevant information. Mind-mapping manually is very laborious as the person has to draw for long periods with great concentration making the process prone to errors. This is our motivation to develop a system that automatically generates mindmaps from the text. Our system takes the input to be in a written or an audio format. Moreover, there is also a facility given to the user to download the mindmap to his local machine where he can refer to the mindmap multiple times. This paper shows the working of our system. The skeleton of the remainder of our paper is as follows. Section II consists of the existing work related to methodologies, tools, techniques, etc. Section III has the proposed solution that contains the system architecture and the flow of the algorithm. Section IV contains the results we have obtained followed by section V where we have defined the future scope of Our project and Section VI where we conclude the paper.

2 Related Work:

In this section, we talk about the numerous methodologies, tools, and algorithms that are used for mind-map generation and the tools and algorithms that we have used in our system for a mind-map generation.

- **Literature related to existing tools:** In this subsection, we talk about the numerous tools that exist which help in mind-map generation

Textstorm: This tool does not require any prior knowledge of the domain that the user is working on. It produces binary predicates as a result after parsing a sentence [3]. For e.g.: The sentence is "Tiger loves to eat Goat". The binary predicate output will be "eat (Tiger, Goat). Thus, these binary predicates are used to represent a relationship between two concepts, in this case, the tiger and the goat. The subjects of the sentence are identified as one concept and the verb is used to connect it to another concept. The resulting output is then given to another tool such as Clouds [4] that requires user intervention to construct a mind-map. The main disadvantage of this system is that it is not automated nor is it capable of constructing a mind-map on its own. It produces results that are taken for further processing to construct a mind map.

Clouds: This tool does not have any previous knowledge of the domain on which the input is based [4]. Its main function is to suggest to the user about the different concepts present in the text and the relationship between these concepts. One limitation of this tool is that the basic concepts in the text have to be fed into the system as inputs.

CmapTools: It is a tool that allows a user to manually construct mind maps. It consists of three main modules, the concept suggester searches and suggests various concepts to the user so that he/she can construct the mind map. Moreover, this tool also has multimedia support as based on these concepts' information is scraped off the web, the suggester that puts forward various multimedia resources and preposition suggestions, and the extender which suggests new topics that could be included in the mind map [5].

The above tools can be classified as tools that help in the semi-automatic construction of concept maps. The major disadvantage of these tools is that they do not reduce human effort and are pretty time-consuming. Moreover, these tools are always dependent on something making them infeasible to use.

The table given below provides information related to tools that construct mind maps automatically and completely on their own.

Table 1: Related Tools

Name of Tool	Description	Disadvantage
Gnosis	This tool that was developed by [6] generates mind maps on the basis of the number of times words have occurred in a sentence i.e., on the basis of the frequency of word occurrence. The results of the system were somewhat positive as it was able to identify relations between two different concepts.	The main disadvantage of this system was that the relations between the concepts were not labelled making it difficult to understand how these concepts are related.
Relex	This system was put forward by [7] and it analyses the sentence so as to extract noun-verb-noun relations as well as the various noun phrases present in the text.	There is no provision for voice input as it only works with text files.
Textract	This tool is developed by Amazon and makes use of Machine Learning Algorithms to extract concepts from a stream of text. The text input can also be in the form of PDFs or tables [8].	The main disadvantages of this system are that it is very expensive and it is not multilingual. Moreover, the amount of text that can be given as input in one time is limited.

Leximancer	This tool makes use of data-mining algorithms to find out the frequency of the words present in the given text. The output is in the form of a graph where the nodes are the main concepts and the edges joining them represent the relations between them. It makes use of Bayes Algorithm to find the relationship between concepts i.e., to calculate how related they are to each other [9].	It does not have any feature for an audio input.
Wordle	This tool is developed by IBM where the user decides how the final results should look like. The result consists of various clouds representing the concepts present in the text. The user can change the designing of the clouds to his liking.	It is very expensive and also has a very few and limited features.

- **Literature related to existing methodologies:**
  - *Two-phase concept map construction:* This method includes two main phases, first is the grade fuzzy association rule phase and the concept map construction phase. The objective of this algorithm is to not identify and take out the concepts present in the text but to identify the pre-requisite relationships that are present among these concepts. Using these pre-requisite relationships, the algorithm generates the mind map [10].
  - *Using a neural network language model (NNLM):* Using this technique an architecture was proposed by [12] where the word vector representation was learned by a neural network having a linear projection layer and a non-linear hidden layer. Following this, another architecture was proposed by [13,14] where the word vectors are learned using a neural network with a single hidden layer which in turn is used to train the NNLM.
  - *M2Gen:* In this algorithm we first start by converting the text into a semantic model. There are three main techniques that help in the above conversion. These techniques are Morphological analysis, semantic analysis, and parse tree analysis. The morphological analysis further consists of Parts-of-speech tagging and lemmatization which help in-text pre-processing. The result of morphological analysis is then parsed by keeping the context-free grammar rules in mind. The result of this is a parse tree which is then given as an input for semantic analysis where operations such as word-sense disambiguation, discourse analysis, and text representation are performed. The final result generated is the semantic model which is used for a mind-map generation [11].
  - *Skip Gram:* This is an unsupervised technique whose main objective is to find words that are somewhat similar to a given word. These words are then given as an input to the neural network that gives the result as probabilities of occurrence of each word in the given text. The result obtained is displayed in the form of a vector of 0's and 1's where 1 means that the word is present in the vocabulary and 0 meaning that it is absent. The training complexity is proportional to

    $$Q = C * (D + (D * V))$$

    Where C = max distance of the words.
  - *Actor-based Mind Map assembler:* In this method the algorithm extracts the subject-verb and object present in the text stream. This method does not make use of a semantic model, instead, it makes use of a syntactic parse tree to generate results. The subject and object are defined as the concepts and the verb is used to describe the relationship between the two subjects [15]. The working of the algorithm is shown in the given figure.

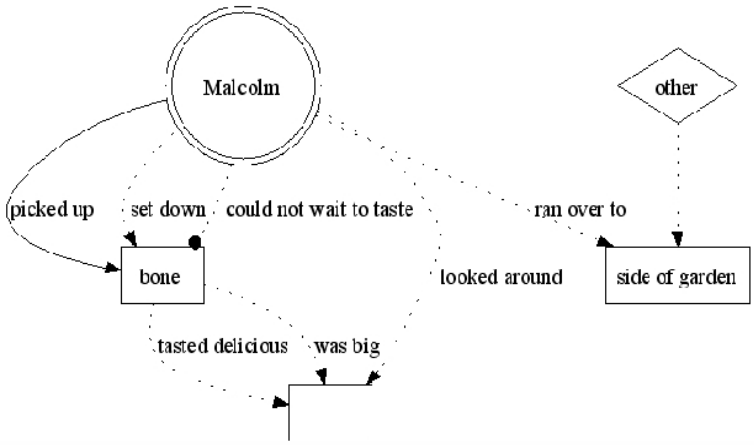

Figure 2 Actor Based Mind Map

3 Proposed Solution:

The user input should be in textual or verbal format. The textual input text can be pasted or typed in the text area. In the case of verbal format, the system will transcribe in real-time using the speech-to-text function once the microphone permission is allowed by the user's machine. This input data will go through the preprocessing functions to get rid of unnecessary data. Furthermore, they will be classified into grammatical categories such as nouns, pronouns, verbs, etc. These recognized entities will now be vectorized using the pre-trained python models of the spacy library.

These vectorized representations allow performing numerical operations that are required to generate the output mind map. These numerical operations are performed to establish relationships among them. These relationships aid in the formation of the mind map by making a tree-like structure. The root node of the tree structure is the central concept of the text.

This structure is then rendered as a Mind Map. This is done with the aid of external graph visualization JavaScript library.

3.1: Proposed System Architecture

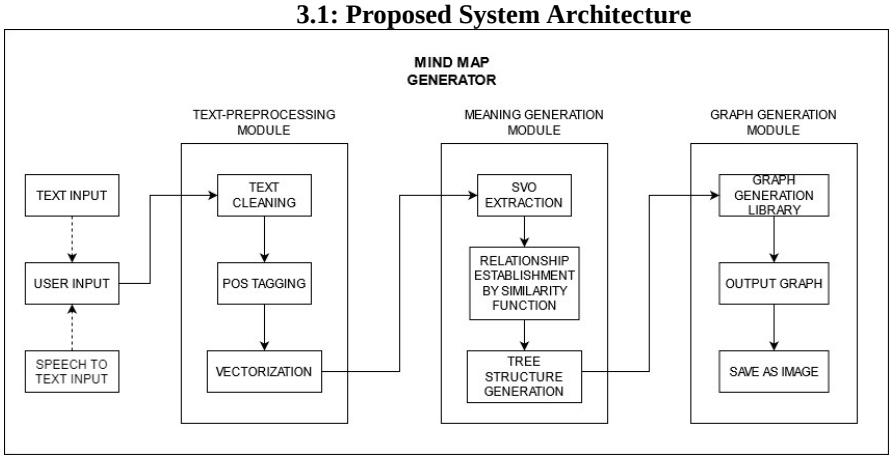

Figure 3 Proposed System Architecture

The proposed system architecture consists of 3 major modules. Each module is further divided into sub-modules for ease of implementation. The three main modules are Text-Preprocessing Module, Meaning Generation Module, and Graph Generation Module. The user can provide textual or verbal input to the system. In the case of verbal input, it is transcribed in real-time. After this, to ensure uniformity, Text-Preprocessing is performed wherein text cleaning, POS tagging, and vectorization is done. Different operations are performed on this pre-processed text to generate meaningful structure. Finally, the data and the metadata help in the generation of the Mind Map.

- **User Input:** Can be verbal or textual input by the user. The speech-to-text feature of the system transcribes the verbal input to text in real-time efficiently.
- **Text-Preprocessing Module:** This is the first module that acts on the input data provided by the user. There are 3 sub-modules
    - *Text-Cleaning:* The input data is raw, unfiltered, and unstructured. Hence, it needs to be filtered out from unnecessary noise. The system removes article numbers (E.g. [5]), percentage values, non-English characters (E.g., è, È), and replaces multiple line breaks with a single.
    - *POS Tagging:* To carry out further operations, each word is given its respective Parts-Of-Speech (POS) tag.
    - *Vectorization:* Mathematical operations cannot be performed on simple text. So, to make text data eligible for calculations, vectorization is performed. 2D representation of every sentence in the input data is done to perform further calculations.
- **Meaning Generation Module:** This module forms the main core of the system. Critical operations that determine the correctness of the output graph are performed here. This module has 3 sub-modules

- *SVO (Subject-Verb-Object) Extraction:* For each sentence in the pre-processed data, the verbs (not auxiliary verbs) are found with the help of POS tagging. Now for this verb, corresponding subjects and objects are found, thus forming an SVO pair. In the case of multiple SVO from a single sentence, all of them are included in the list where SVOs are stored. The final Mind Map graph shall be formed primarily from this list.
- *Relationship establishment using similarity function:* To make a connected undirected mind map, relationships between pairs of SVOs are to be determined, this is done using cosine similarity as the data is already vectorized. Two SVO pairs are connected if and only if the cosine similarity value is greater than or equal to the threshold value 0.5. Cosine similarity is used we only need if the value is above or below the threshold and not the magnitude difference. An adjacency matrix is formed that represents the connections for the final Mind Map.
- *Tree Structure Generation:* To make the adjacency list convey information to the JavaScript rendering library, the data and the metadata are used and a cytoscape.js compatible JSON structure is generated.

- **Graph Generation Module:** This is the final module that is responsible for the correct rendering and output display.
  - *Graph Generation Library:* The JSON data from the previous step is now processed by Cytoscape.js which is a free open-source graph visualization JavaScript Library. It is responsible to generate the Mind Map graph with all the necessary connections as specified in the previous module's calculations.
  - *Output Graph:* This is the final rendered Mind Map of the input text data that was provided by the user.
  - *Save As Image:* The system allows downloading the generated Mind Map in JPEG format and saving it to the machine at a specified location.

3.2 Proposed Algorithm

Algorithm 1: Text Preprocessing:

text = user input {can be textual or verbal}
new_text = text_cleaning(text) {perform text cleaning on the user input. Remove percentage values, non-English characters, line breaks and article numbers}
Perform POS_tagging(new_text) {Performing Parts-Of-Speech tagging on the cleaned text. Spacy library aids in this process}
Perform vectorization(new_text) {Vectorizing the text in 2D space in order to perform arithmetic operations}
DOC = new_text {storing this pre-processed text in a document}

Algorithm 2: SVO extraction:

```
SVOs = []
FOR verb in DOC and verb != auxiliary verb
   Check for negative verb.
   FIND subject(s) corresponding the verb v
   IF len (subjects > 0):
      FIND objects
      FOR subjects
         FIND objects
Return SVOs
```

For each sentence in the DOC, we find the verb (other than auxiliary verb). Now corresponding to this verb, its subject(s) and object(s) are extracted. If there are multiple subjects present, we find the corresponding objects for the multiple subjects. This is the main idea of SVO extraction. All of the pairs are stored in the SVO list.

Algorithm 3: Adjacency list of edge:

```
adjacency_list = {}
FOR i in range (0, len (SVOs)):
      FOR j in range (i+1, len (SVOs))
            IF similarity (SVOs[i], SVOs[j] > threshold)
                  adjacency_list.append(SVOs[i],SVOs[j])
```

Each SVO pair is tested for similarity with all the other SVO pairs. If the cosine similarity value is greater than or equal to 0.5, then and only then the pair is appended in the adjacency list. At the end, an adjacency list connecting the similar SVOs is formed.

Algorithm 4: Graph Generation:

Cytoscape.js compatible JSON format <- adjacency_list + metadata
Cytoscape.js Graph Rendered Output <- JSON data
The adjacency list and metadata together get converted to a cytoscape.js compatible JSON structure. This then produces the output mind map graph.

4 Testing and Results:

Table 2: Test case and results

Test Case No.	Description	Expected Outcome	Actual Outcome	Result
1.	Short input of fewer than 70 words or 7 sentences is provided as text input by the user.	If the text is relevant and focused on a central idea, a Mind Map should be generated.	A mind map of the central idea as a root node is generated.	Pass
2.	The user provides input via the speech-to-text feature by speaking out every word.	The system should generate accurate transcripts of the speech input.	Transcripts were generated accurately for the verbal input provided.	Pass
3.	The user provides speech input of numbers, special characters, and punctuations.	The system should convert the said input to their respective representation. (Ex: 'at the rate' should become '@').	The system correctly identifies numbers, symbols, and punctuations from the verbal input and generates transcripts.	Pass
4.	Large input of 2000 words is provided.	If the text is relevant and focused on a central idea, a Mind Map should be generated.	The system generates a mind map but does not cover everything of the input. The map generated is just satisfactory.	Fail
5.	Download the generated map as JPEG.	The system should ask for the location on the local disk to download the image.	The system allows you to download the map as a JPEG image at the specified destination.	Pass

Results:

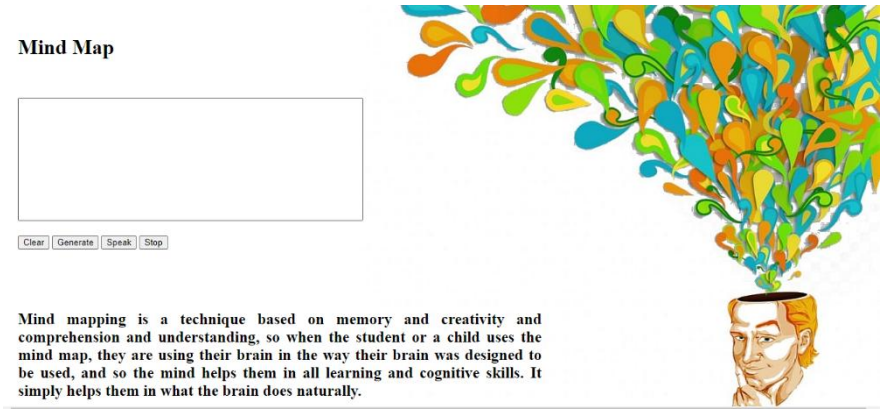

Figure 4 Input Page / Landing Page

The above image (figure 4) shows the input page of the system. The user may paste or type text in the text area as shown in figure 4. For verbal input, the user must first click on "Speak" button, allow browser permission for accessing the microphone. Now the user can provide the verbal input. To stop the recording, the user must hit the "Stop" button. All the transcribed text would appear in the text-area in real time. The "Clear" button is used to clear out the text in the text area. To generate the mind map, click on "Generate" after providing the input.

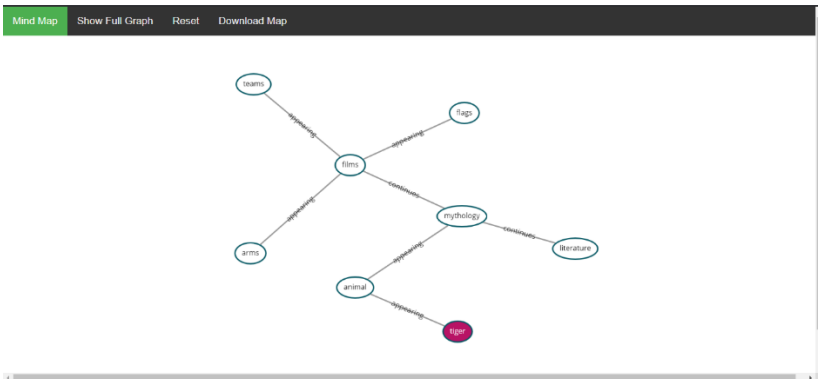

Figure 5 Output Page

The above image (figure 5) shows the output page consisting a MindMap. The node colored in maroon shows the central node (central idea) of the input text. The SVO pairs are connected according to the adjacency list generated after performing numerical operations. All the nodes can be dragged around as per the user. The text on the edge is the action verb whilst the nodes are subjects and objects. The "Show Full Graph" shows the entire graph where the number of connections limit is not considered. "Reset" button resets the graph to its original state. The "Download Map" allows to save the MindMap as JPEG.

Observational results

- The system was able to generate an appropriate mind map irrespective of the text topic considered. However, the best results are generated when the input focuses on a single central topic. Duality or vague text seems to confuse the system to find the central idea (root node). This hampered the meaningful generation of mind map as the central idea is not what the user expects most of the time.
- Precise text with formal information yields a better result and conveys maximum information. In contrast, casual text containing information such as description produced low quality mind maps.
- For speech input, the system recognizes the verbal audio and transcribes it to text accurately. However, the punctuations, special characters too must be spoken along with the words.

  For example: Textual input: There are dogs, cats and rats.
  Corresponding Verbal Input: There are dogs *comma* cats and rat *full stop*
  The system is not capable of providing punctuations on its own hence they must be pronounced explicitly.
- The response time of the system to generate a mind map is mainly dependent on the number of input words. For text with length of 2500 words and above produced understandable mind maps. They consumed about 4 to 6 seconds to produce the final output. On the other hand, relatively smaller text with words up to 1500 responded with 1 to 3 seconds. Despite this, computation speed also varies from system to system.

5. Future scope:

In this section, we explain how we envision the expansion of our project. We aim to further improve the project by making it multilingual i.e the system can make mindmaps not only from the text given in English but also it can make mindmaps from textual data in other languages such as Hindi, French, etc. The technology used for converting text in English to any other language is called neural machine translation which is currently a primary area of research. Furthermore, we will try to add a pictorial element to the graph by replacing all the nodes with the respective images they denote. Finally, we ain to integrate with a cloud service such as Google Cloud Platform(GCP) where it can directly get hold of the data stored in their cloud suite(google docs, google drive, etc)

6. Conclusion

The system that we have developed helps to enhance productivity by converting huge volumes of textual data into mind maps that can easily be understood and remembered by an individual. Moreover, we have added features such as the speech-to-text module that allows the user to dictate the information he wants to convert into mind maps thereby easing the workload on their shoulders. Furthermore, we have deployed the entire system as a web application so that the user can use it anytime he wants and can download the image of the mind map to his system if needed. The algorithm that we have used does not only make use of semantics to understand the relationship between concepts but also makes use of grammatical approaches. Finally, the results that we have obtained show promise and we aim to further improve the system.

7. References

1. https://techjury.net/blog/how-much-data-is-created-every-day/gref
2. Marshall KS, Crawford R, Jensen D (2016) Analogy seeded mind[1]maps: a comparison of verbal and pictorial representation of analogies in the concept generation process, vol 7. In: 28th inter[1]national conference on design theory and methodology, ASME, p V007T06A010
3. A. O Alves, F. C. Periera, and A. Cardoso. Automatic reading and learning from text. In ISAI'2001: In Proceedings of the International Symposium on Artificial Intelligence, pages 302–310, December 2001
4. Francisco C. Pereira and Am´ılcar Cardoso. Clouds: A module for au[1]tomatic learning of concept maps. In Michael Anderson, Peter Cheng, and Volker Haarslev, editors, Diagrams, volume 1889 of Lecture Notes in Computer Science, pages 468–470. Springer, 2000.
5. David B. Leake, Ana Maguitman, Thomas Reichherzer, Alberto J. Ca˜nas, Marco Carvalho, Marco Arguedas, Sofia Brenes, and Tom Es[1]kridge. Aiding knowledge capture by searching for extensions of knowl[1]edge models. In K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture, pages 44–53, New York, NY, USA, 2003. ACM Press

6. Brian R. Gaines and Mildred L. G. Shaw. Using knowledge acquisition and representation tools to support scientific communities. In AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1), pages 707–714, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.
7. Ryan Richardson, Ben Goertzel, Edward A Fox, and Hugo Pinto. Automatic creation and translation of concept maps for computer science related theses and dissertations. In Proceedings of 2nd Concept Mapping Conference, pages 160–163. Citeseer, 2006.
8. https://aws.amazon.com/textract/
9. Andrew E. Smith. Leximancer manual (Version 2.2). University of Queensland, 2005.
10. Pei-Chi Sue, Jui-Feng Weng, Jun-Ming Su, and Shian-Shyong Tseng. A new approach for constructing the concept map. icalt, 0:76–80, 2004.
11. M. Abdeen, R.El-Sahan, A.Ismaeil, S.El-Harouny, M.Shalaby, M.C.E. Yagoub, "Direct Automatic Generation of Mind Maps from Text with M2Gen", in Proceeding of IEEE Toronto International Conference Science and Technology for Humanity, pp. 95-99, Toronto, Canada, 2009.
12. Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Ma[1]chine Learning Research, 3:1137-1155, 2003.
13. T. Mikolov. Language Modeling for Speech Recognition in Czech, Masters thesis, Brno University of Technology, 2007.
14. T. Mikolov, J. Kopecky, L. Burget, O. Glembek and J. Cernocky. Neural network based lan- ´ guage models for higly inflective languages, In: Proc. ICASSP 2009.
15. C.Brucks, C.Schommer, "Assembling Actor-based Mind-Maps from Text Streams", Master Thesis, University of Luxembourg, Department of Computer Science and Communication, 2008.